REMARKS

This application has been carefully reviewed in light of the Office Action dated May 3, 2006. Claims 2 and 3 have been cancelled herein, without prejudice or disclaimer of subject matter. Claims 1, and 4 to 30 remain in the application, of which claims 1, 4 to 7, 9, 11 to 15, 18 to 23, 25, 26 and 29 have been amended herein. Claims 1, 11 and 21 are the independent claims. Reconsideration and further examination are respectfully requested.

In the Office Action, claims 1 to 3, 10 to 13, 21, and 27 to 30 were rejected under 35 U.S.C. § 102(b) over U.S. Patent No. 5,784,699 ("McMahon"); and claims 4 to 9, 14 to 20, and 22 to 26 were rejected under 35 U.S.C. § 103(a) over McMahon in view of U.S. Patent No. 5,930,827 ("Sturges"). In response, the independent claims have been amended herein to further clarify the features that *i)* a block of memory is divided into a plurality of frames, with each of the plurality of frames operable to store an indexing structure associated with a single attribute of a data record, and *ii)* each of the plurality of frames is divided into a plurality of instances, with each of the plurality of instances operable to store an index node of the indexing structure.

The present disclosure generally relates to memory management. A block of memory is divided into a plurality of frames, with each of the plurality of frames operable to store an indexing structure associated with a single attribute of a data record. Each of the plurality of frames is divided into a plurality of instances, with each of the plurality of instances operable to store an index node of the indexing structure. Unused instances are then identified.

Referring to particular claim language, independent claim 1 recites a frame handler for application-level memory management. The frame handler includes an associated block of memory divided into a plurality of frames, with each of the plurality of frames operable to store an indexing structure associated with a single attribute of a data record, and each of the plurality of frames divided into a plurality of instances, with each of the plurality of instances operable to store an index node of the indexing structure. The frame handler also includes a data structure identifying the unused instances within each of the plurality of frames, and an application interface operable to receive a request for an unused instance from a software application. The frame handler is operable to identify an unused instance in response to a request received by the application interface.

Independent claim 11 recites a method for allocating memory in a computer system. The method includes outputting a request from an application to an operating system for allocation of a block of memory by the operating system to the application, accessing the block of memory for the application, and dividing the block of memory into a plurality of frames, with each of the plurality of frames operable to store an indexing structure associated with a single attribute of a data record. The method also includes dividing each of the plurality of frames into a plurality of instances, with each of the plurality of instances operable to store an index node of the indexing structure, and maintaining a data structure identifying the unused instances within each of the plurality of frames.

Independent claim 21 recites a method, including accessing a block of memory for an application, and dividing the block of memory into a plurality of frames, including first and second frames, with each of the plurality of frames operable to store an indexing structure associated with a single attribute of a data record. The method also includes dividing each of the plurality of frames into a plurality of instances, including first and second lists of instances, with each of the plurality of instances operable to store an index node of the indexing structure. Additionally, the method includes assigning a first identifier that is associated with the first frame to a first frame node, linking the first list of instances to the first frame node, and assigning a second identifier that is associated with the second frame to a second frame node. The method also includes linking the second list of instances to the second frame node, and constructing a data structure using a plurality of nodes including the first node and the second node. Furthermore, the method also includes selecting available instances within each of the plurality of frames using the data structure, via an application.

The applied art is not seen to disclose, teach, or to suggest the foregoing features recited by the independent claims. In particular, McMahon is not seen to disclose at least the features that *i)* a block of memory is divided into a plurality of frames, with each of the plurality of frames operable to store an indexing structure associated with a single attribute of a data record, and *ii)* each of the plurality of frames is divided into a plurality of instances, with each of the plurality of instances operable to store an index node of the indexing structure.

McMahon describes a computer system which implements a memory allocator that employs a data structure to maintain an inventory of dynamically allocated memory available to

receive new data. *See* McMahon, col. 4, ln. 58 to col. 5, ln. 8; and Abstract. Although the Office Action, at page 11, alleges that the free list discloses the feature of identifying unused *instances*, where *instances* are within a block of memory, the applicants again respectfully disagree. Clearly, the passage beginning at column 5, line 25, is seen to describe using a free list to search for an available memory *block*, and not an available *instance* within a block. In particular, the passage states (emphasis added):

> In operation, the dynamic memory allocator 50 receives requests from a software program. As shown in block 110, the dynamic memory allocator 50 rounds the memory request size to the nearest bin size in accordance with the predetermined rounding factor. The dynamic memory allocator 50 *searches a free list*, corresponding to the bin size, *for an available* ***memory block*** as shown in block 120. If a ***memory block*** *is available* for the appropriate bin size, then the dynamic memory allocator 50 *assigns a* ***memory block*** utilizing the pointer on the free list as shown in blocks 130 and 135. As shown in blocks 130 and 140 on FIG. 2, *if a* ***memory block*** *is not available* in the specified bin, then the dynamic memory allocator 50 searches for an *available* ***memory block*** from a free list that corresponds to the next largest bin size. (McMahon, col. 5, ll. 25 to 39).

Accordingly, the cited passage is not seen to describe searching for anything other than an available memory *block*, and is certainly not seen to describe the feature of identifying available, or unused *instances* within a memory block. Furthermore, McMahon is not seen to describe, nor does the Office Action even assert that McMahon describes, the newly clarified features of the independent claims, that *i)* a block of memory is divided into a plurality of frames, with each of the plurality of frames operable to store an indexing structure associated with a single attribute of a data record, and *ii)* each of the plurality of frames is divided into a plurality of instances, with each of the plurality of instances operable to store an index node of the indexing structure.

Accordingly, based on the foregoing amendments and remarks, independent claims 1, 11 and 21 are believed to be allowable over the applied references. The remaining rejected claims in the application are each dependent on these independent claims and are believed to be allowable for at least the same reasons. Because each dependent claim is deemed to define an additional aspect of the invention, individual consideration of each on its own merits is respectfully requested.

No other matters being raised, it is believed that the entire application is fully in condition for allowance and such action is courteously solicited.

No fees are believed to be due at this time. Please apply any other charges or credits to deposit account 06 1050.

Respectfully submitted,

Date: June 28, 2006

David E. A. Jordan
Reg. No. 50,325

Fish & Richardson P.C.
1425 K Street, N.W.
11th Floor
Washington, DC 20005-3500
Telephone: (202) 783-5070
Facsimile: (202) 783-2331

40346124.doc